S-Walk: Accurate and Scalable Session-based Recommendation with Random Walks

Minjin Choi
Sungkyunkwan University
Republic of Korea
zxcvxd@skku.edu

Jinhong Kim
Naver Corp.
Republic of Korea
jinhong.kim93@navercorp.com

Joonseok Lee
Google Research,
Seoul National University
United States, Republic of Korea
joonseok2010@gmail.com

Hyunjung Shim
Yonsei University
Republic of Korea
kateshim@yonsei.ac.kr

Jongwuk Lee*
Sungkyunkwan University
Republic of Korea
jongwuklee@skku.edu

ABSTRACT

Session-based recommendation (SR) predicts the next items from a sequence of previous items consumed by an anonymous user. Most existing SR models focus only on modeling *intra-session* characteristics but pay less attention to *inter-session* relationships of items, which has the potential to improve accuracy. Another critical aspect of recommender systems is computational efficiency and scalability, considering practical feasibility in commercial applications. To account for both accuracy and scalability, we propose a novel session-based recommendation with a random walk, namely *S-Walk*. Precisely, S-Walk effectively captures intra- and inter-session correlations by handling high-order relationships among items using random walks with restart (RWR). By adopting linear models with closed-form solutions for transition and teleportation matrices that constitute RWR, S-Walk is highly efficient and scalable. Extensive experiments demonstrate that S-Walk achieves comparable or state-of-the-art performance in various metrics on four benchmark datasets. Moreover, the model learned by S-Walk can be highly compressed without sacrificing accuracy, conducting two or more orders of magnitude faster inference than existing DNN-based models, making it suitable for large-scale commercial systems.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**; *Expert systems.*

KEYWORDS

Collaborative filtering; Session-based recommendation; Random walks; Closed-form solution

*Corresponding author

WSDM '22, February 21–25, 2022, Tempe, AZ, USA.

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498464

ACM Reference Format:
Minjin Choi, Jinhong Kim, Joonseok Lee, Hyunjung Shim, and Jongwuk Lee. 2022. S-Walk: Accurate and Scalable Session-based Recommendation with Random Walks. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488560.3498464

1 INTRODUCTION

Modern recommender systems (RS) are indispensable for addressing the enormous information overload in various real-world applications, such as e-commerce platforms and online multimedia platforms, *e.g.*, Amazon and Alibaba, YouTube, Netflix, and Spotify. Classical recommender systems [12, 15, 19, 44] usually assume that user accounts and users' long-term interactions are available. However, this assumption rarely holds. The user may not login, or multiple users share the user account, *e.g.*, family members. The user may also exhibit different behaviors depending on the context. Thus, it is necessary to provide personalized recommendations without explicit user information.

Recently, session-based recommendation (SR) [3, 9, 22, 42, 51] has gained considerable attention for predicting the next items from the sequential behavior consumed by an anonymous user. Unlike conventional RS, SR relies only on the users' actions in an ongoing session. This setting of SR is well-suited for real-world scenarios but inherently results in a severe data sparsity problem. To address this issue, it is essential to understand the unique characteristics of sessions, that is, *intra-session* properties. First, the items within a session are coherent with the user's hidden intent, *e.g.*, a list of products in the same category, referred to as *item consistency* (or *long-term dependency*). Second, some items are strictly consumed in chronological order, namely *sequential dependency* (or *short-term dependency*), *e.g.*, consecutive episodes of a TV series. Lastly, the user can repeatedly consume the same items, called *repeated item consumption*, *e.g.*, user's favorite tracks.

Most SR models utilize deep neural networks (DNNs) to learn intra-session relationships. Recurrent neural networks (RNNs) [16–18] and attention mechanisms [31, 32] have been used to model the sequential dependency of items. Recently, graph neural networks (GNNs) [1, 13, 40, 41, 50, 53, 54] have been used to effectively

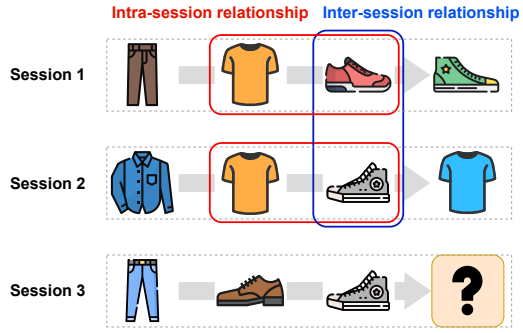


Figure 1: Illustration of intra- and inter-session relationships. The red shoes and the gray shoes (in blue box) do not appear within a single session, but they are correlated because both are related to the orange shirt (in red boxes).

represent both item consistency and sequential dependency. Unfortunately, they suffer from performance degradation when dealing with complex and long sessions, where it is difficult to understand user intent. For this, *inter-session* relationships among items are valuable clues. Figure 1 describes intra-session and inter-session relationships; some items do not occur within a single session but share their neighboring items for multiple sessions, implying potential item correlations. Several studies [36, 50, 52] have attempted to consider inter-session relationships using neighboring sessions or global item graphs. However, they incur substantial computational costs and are infeasible for large-scale commercial systems.

High-capacity DNN models have achieved state-of-the-art performances but usually require heavy computational overhead for runtime speed and memory consumption. Although competing for the best performance is a reasonable mission for most research problems, computational efficiency and scalability are also critical factors for dealing with practical restrictions in commercial recommender systems. [11, 33] suggested neighborhood-based models for session-based recommendations. Owing to their simplicity, they are highly scalable. Moreover, [34, 35] reported that the neighborhood-based models achieved comparable performances to DNN-based models on several benchmark datasets.

Our primary objective is to design an SR model that accounts for both accuracy and scalability. To this end, we propose a novel session-based recommendation with a random walk, namely *S-Walk*, (i) exploiting intra- and inter-session relationships among items to improve accuracy, and (ii) supporting cost-effective, real-time performance at scale.

(i) Whereas the basic SR learns hidden patterns between items only within a session, S-walk additionally introduces *global item graphs*, modeling item-item relationships across all sessions. By applying *random walks with restart (RWR)* on this graph, a random surfer can jump from one item to another adjacent item by traversing the item graph or restart from an arbitrary item in the current session. Therefore, S-Walk can capture high-order correlations among items using multi-hop connections on an item graph. S-Walk can exploit the local patterns within a session and global patterns involving the same items from other sessions.

(ii) Recently, linear item-item models [23, 45–47] have shown competitive performance in conventional RS. Motivated by their success, we devise linear item models to build two probability matrices, *item transition* and *item teleportation* matrices, to formulate the stochastic process of RWR. The item transition matrix captures the sequential dependency of items, generalizing a Markov chain model on items. The item teleportation matrix reflects the restart probability, allowing various items to participate in the model training depending on the ongoing session. Instead of having arbitrary items for restart, we utilize the co-occurrence relationship among items. The items that co-occurred with the current session items are used for restart. Notably, training our linear models is highly efficient and scalable because they have closed-form solutions whose computational complexity is determined by the number of items, independent of the number of sessions or user actions.

To summarize, the key advantages of S-Walk are as follows: (i) It can effectively capture inter- and intra-session relationships via RWR. (ii) Without complicated model tuning, it is highly efficient and scalable owing to the closed-form solution of linear models. (iii) It achieves competitive or state-of-the-art performance in various metrics (*i.e.*, HR, MRR, recall, and MAP) on four benchmark datasets (*i.e.*, YouChoose, Diginetica, RetailRocket, and NowPlaying). (iv) The model learned by S-Walk can be highly compressed without sacrificing accuracy, supporting fast inference time.

2 PRELIMINARIES

Notations. Given a set of sessions $\mathcal{S} = \{s^{(1)}, \ldots, s^{(m)}\}$ over a set of items $\mathcal{I} = \{i_1, \ldots, i_n\}$, an arbitrary session $s \in \mathcal{S}$ is represented by a sequence of items $s = (s_1, s_2, \ldots, s_{|s|})$ with a length $|s|$. Here, $s_j \in \mathcal{I}$ is the j-th consumed item, *e.g.*, clicked, watched, or purchased. For simplicity, $s \in \mathcal{S}$ is represented by a binary vector $\mathbf{x} \in \{0,1\}^n$, where $\mathbf{x}_k = 1$ if i_k is consumed, and $\mathbf{x}_k = 0$ otherwise. By stacking m sessions, let $\mathbf{X} \in \mathbb{R}^{m \times n}$ denote a session-item interaction matrix, where m is the number of sessions. As a straightforward variant, the binary value in $\mathbf{x}_k$ can be converted to real values to quantify the importance of items within a session.

Problem statement. Given a sequence of items previously consumed by an anonymous user in a session, session-based recommendations predict the next items that the user is most likely to consume. Formally, a session-based recommender model takes a session $s = (s_1, \ldots, s_j)$ as input and returns a ranked list of top-N candidate items as the recommended next items $(s_{j+1}, \ldots s_{|s|})$. Note that this is more generalized (and challenging) than predicting the next single item s_{j+1}, which has been used in existing work. (See Section 4 for the generalized evaluation metrics for this setting.)

Random walk models. The key concept behind random walk models is to reflect the direct and transitive relations among items. Conventional recommender models using random walks [6–8, 20, 37, 38, 55] are based on a user-item bipartite graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E})$, where $\mathcal{U}$ and $\mathcal{I}$ are a set of users and items. Each edge $e \in \mathcal{E}$ represents the relationship between a user and an item. Thus, the core part of the random walk model is to determine a transition probability matrix to compute *proximity* scores for items.

In general, there are two possible solutions for computing the proximity scores of items. First, we can utilize the k-step landing probability distribution of a random walker. Starting from a source user u, the proximity scores for all items can be computed as $\mathbf{uR}_{(k)}$,

where $\mathbf{u}$ is the user vector, and $\mathbf{R}_{(k)}$ is the k-step transition probability matrix. Although it is simple yet effective, existing studies [6, 7] are vulnerable to popularity bias; popular items tend to have high proximity scores as k increases (*e.g.*, $k \geq 3$), thereby degrading the recommendation quality.

As an alternative, we can compute the stationary distribution of *random walks with restart (RWR)*, which is well-known as personalized PageRank [26]. It is effective for capturing high-order relationships among vertices. Because RWR leverages *restart* in addition to *sequential transition*, it can alleviate the problem of popularity bias, concentrating on the central node in the k-step. For this reason, we adopt RWR for session-based recommendations. (See Section 5 for empirical comparisons between the two methods, k-step and RWR.)

Adopting the random walk in session-based recommendations has the following advantages: (i) The random walk model utilizes high-order item correlations across sessions. Because sessions are usually sparse by nature, it is useful for alleviating the data sparsity problem by capturing profound relationships among items. (ii) Compared to GNN-based SR models [13, 40, 41, 53, 54], it is efficient without requiring complicated hyper-parameter tuning.

Linear item-item models. Given the session-item matrix $\mathbf{X}$, the goal of linear models [39, 46] is to estimate the item-item similarity matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$. As a pioneering work, SLIM [39] formulated a linear model subject to the constraint that all entries in $\mathbf{B}$ are non-negative and zero diagonal.

$$\begin{aligned} \underset{\mathbf{B}}{\operatorname{argmin}} & \|\mathbf{X} - \mathbf{X} \cdot \mathbf{B}\|_F^2 + \lambda_1 \|\mathbf{B}\|_1 + \lambda_2 \|\mathbf{B}\|_F^2 \\ & \text{s.t. } \mathtt{diag}(\mathbf{B}) = 0, \quad \mathbf{B} \geq 0, \end{aligned} \tag{1}$$

where $\|\cdot\|_1$ and $\|\cdot\|_F$ are the entry-wise ℓ_1-norm and the matrix Frobenius norm, respectively, λ_1 and λ_2 are the regularization coefficients, and $\mathtt{diag}(\mathbf{B}) \in \mathbb{R}^n$ denotes the vector with the diagonal elements of $\mathbf{B}$. Although SLIM [39] shows competitive accuracy, it suffers from high computational training cost.

Recently, EASE^R [46] and its variants [45, 47] only consider the zero-diagonal constraint by removing the non-negativity of $\mathbf{B}$ and ℓ_1-norm constraints from Eq. (1):

$$\underset{\mathbf{B}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{X} \cdot \mathbf{B}\|_F^2 + \lambda \cdot \|\mathbf{B}\|_F^2 \quad \text{s.t. } \mathtt{diag}(\mathbf{B}) = 0. \tag{2}$$

Owing to this simpler formulation, EASE^R is solved by the closed-form equation via Lagrange multipliers:

$$\hat{\mathbf{B}} = \mathbf{I} - \hat{\mathbf{P}} \cdot \mathtt{diagMat}(\mathbf{1} \oslash \mathtt{diag}(\hat{\mathbf{P}})), \tag{3}$$

where $\hat{\mathbf{P}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}$. Here, $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones, and $\oslash$ denotes the element-wise division. Let $\mathtt{diagMat}(\mathbf{x})$ denote the diagonal matrix expanded from the vector $\mathbf{x}$. (See [46] for the detailed derivation of the closed-form solution.)

Although inverting the regularized Gram matrix $\mathbf{X}^\top \mathbf{X}$ is the computational bottleneck for large-scale datasets (*i.e.*, the time complexity is $O(|\mathcal{I}|^{2.376})$ with the Coppersmith-Winograd algorithm), the closed-form solution is advantageous in terms of efficiency. The training complexity of EASE^R [46] is proportional to the number of items, which is usually much smaller than the number of sessions ($n \ll m$). Besides, the linear model is beneficial to accelerate the inference because computing top-N recommended items is simply done by single matrix multiplication. Most recently, SLIST [5] reported competitive accuracy with linear models for the session-based recommendation. Although SLIST [5] tackled various characteristics of session data, it did not consider inter-session relationships. In contrast, we devise linear models with random walks, taking both intra- and inter-session correlations into account.

3 S-WALK: THE PROPOSED MODEL

In this section, we propose a novel session-based recommendation with a random walk, namely *S-Walk*. While existing random-walk-based recommender models [6–8, 20, 37, 38, 55] are based on a user-item bipartite graph, it is non-trivial to adopt them for session-based recommendations. Since user information is unavailable, we rely only on item information to learn underlying patterns. Furthermore, the session-item matrix is extremely sparse.

To address these issues, we first present the overall architecture of S-Walk using *global item graphs* (Section 3.1). Intuitively, walking on a global item graph can describe inter-session relationships among items because the walker can move from an item of the current session to the items of other sessions. Then, we develop two linear models to build a transition matrix and a teleportation matrix, used in S-Walk (Sections 3.2–3.3). Note that these linear models can be replaced by others as long as they are efficient and scalable. Notably, our linear models satisfy the desirable condition for efficiency and scalability. Finally, we explain model training and inference of S-Walk (Section 3.4).

3.1 Model Architecture

Figure 2 overviews the S-Walk model. Given a session-item interaction matrix $\mathbf{X}$, we first design two models for item transition and teleportation to capture different characteristics of sessions (the blue and orange box in Figure 2, respectively). We then constitute a final global item graph using *random walk with restart (RWR)*.

Specifically, each model produces its own relevance matrix over the transition graph $\mathcal{G}_\mathbf{R} = (\mathcal{I}, \mathcal{E}_\mathbf{R})$ and the teleportation graph $\mathcal{G}_\mathbf{T} = (\mathcal{I}, \mathcal{E}_\mathbf{T})$, where each node corresponds to an item and an edge indicates the relevance between a pair of items. The *transition matrix* $\mathbf{R}$ is the adjacency matrix of $\mathcal{G}_\mathbf{R}$ which encodes sequential dependency and repeated item consumption in a session. On the other hand, the *teleportation matrix* $\mathbf{T}$ is the adjacency matrix of $\mathcal{G}_\mathbf{T}$, which captures item consistency in the session. Introducing the two matrices captures different intra-session relationships among items, but they do not address the inter-session relationship.

By adopting the RWR using the two graphs, where a random walker jumps from one node to another or restarts on an arbitrary node regardless of her current position, we intend to consider the inter-session relationship, capturing high-order relationships among items, *i.e.*, multi-hop connections on the item graphs. Conceptually, the RWR on the two item graphs, $\mathcal{G}_R$ and $\mathcal{G}_T$, can be thought of as tossing a biased coin that yields the head with probability α:

(1) If the coin is head (with a probability of α), the walker moves to one of the items adjacent to the current item through the transition matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$.
(2) If the coin is tail (with a probability of $1 - \alpha$), the walker restarts on one of the items adjacent to the start item through the teleportation matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$.

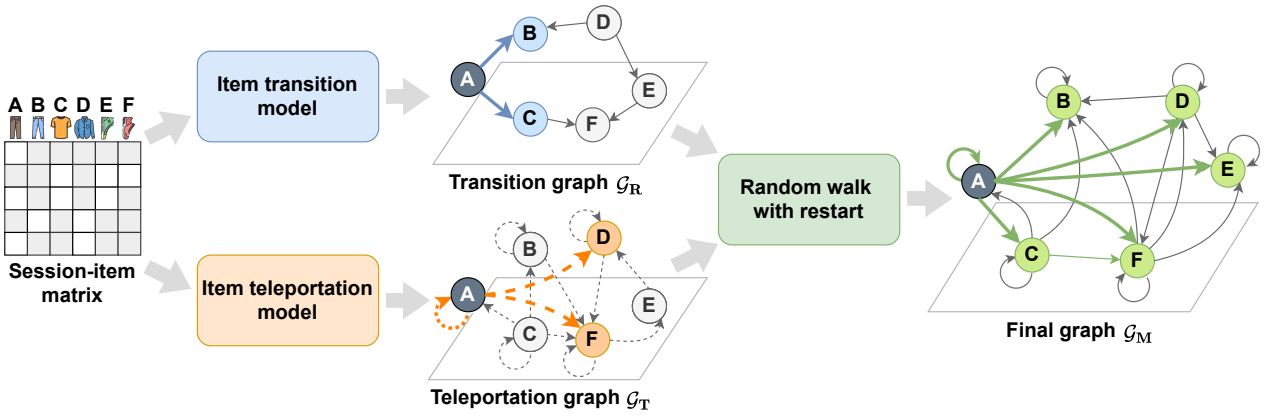


Figure 2: The overall architecture of S-Walk. Given a session-item matrix, two different linear models build the transition graph and the teleportation graph by adjacency matrices R and T, using Eq. (7) and Eq. (10), respectively. Then, in Eq. (12), the random walk with restart is used to build the final graph with adjacency matrix M, capturing high-order relationships.

The random walk using these two matrices is a stochastic process, which can also be seen as a Markov chain on items over a homogeneous discrete time. Formally, we formulate the RWR as follows:

$$\mathbf{x}_{(k+1)} = \alpha \mathbf{x}_{(k)} \mathbf{R} + (1-\alpha)\mathbf{x}_{(0)}\mathbf{T}, \quad \text{where} \quad k = 0, \ldots, \infty. \tag{4}$$

Here, α is the damping factor that controls the proportion of the random walk and the restart. $\mathbf{x}_{(0)}^{\top} \in \mathbb{R}^n$ is the initial item vector and $\mathbf{x}_{(k)}^{\top} \in \mathbb{R}^n$ is the updated proximity score for items after the k-th step. As the k increases, $\mathbf{x}_{(k)}$ converges to limited distribution. Through the RWR, we obtain stationary probabilities that the random walker lands on each node, expressed as the green graph in Figure 2. Finally, we generate the recommendation list using the final graph $\mathcal{G}_{\mathrm{M}} = (\mathcal{I}, \mathcal{E}_{\mathrm{M}})$.

In this process, we devise linear models for the two models, taking the following advantages: (i) they achieve comparable performance without complicated tuning, and (ii) training and inference are much faster than DNN-based session recommender models [13, 16–18, 31, 32, 40, 41, 53, 54].

3.2 Item Transition Model

Fist, we develop a linear transition model to build the item transition matrix $\mathbf{R}$. As a natural way of representing the transition of item sequences, we introduce *partial session representations*. A session s is divided into two sub-sessions, *past* and *future*, according to each time step $t = 1, ..., |s|$. The past partial session consists of items consumed before the t-th item, *i.e.*, $\mathbf{s}_{1:t-1} = \{s_1, \ldots, s_{t-1}\}$. The future partial session consists of items consumed at or after the t-th item, *i.e.*, $\mathbf{s}_{t:|s|} = \{s_t, \ldots, s_{|s|}\}$. For each time $t = 2, ..., |s|$, we produce $|s| - 1$ past and future partial session pairs. By stacking $|s| - 1$ pairs for all $s \in \mathcal{S}$, we build two matrices, the past session matrix $\mathbf{Y} \in \mathbb{R}^{m' \times n}$ and future session matrix $\mathbf{Z} \in \mathbb{R}^{m' \times n}$, where m' is the number of all partial sessions, *i.e.*, $m' = \sum_{i=1}^{m} (|s^{(i)}| - 1)$. Finally, the item transition matrix $\mathbf{R}$ is learned with two matrices $\mathbf{Y}$ and $\mathbf{Z}$ according to the partial session representation.

To represent the temporal proximity of items, we adjust the weights of items in $\mathbf{Y}$ and $\mathbf{Z}$. As adopted in [5, 11], we consider the position gap between two items as the weight of items within a session.

$$w_{\mathrm{pos}}(i, j, s) = \exp\left(-\frac{|p(i,s) - p(j,s)|}{\delta_{\mathrm{pos}}}\right), \tag{5}$$

where δ_{pos} is the hyper-parameter that controls the position decay in partial sessions, and $p(i, s)$ is the position of item i in the session s. In this way, we decay the relevance between items i and j as they get farther away.

Formally, the item transition model is formulated as

$$\underset{\mathbf{B}^{\mathrm{tran}}}{\operatorname{argmin}} \left\|\mathbf{Z} - \mathbf{Y} \cdot \mathbf{B}^{\mathrm{tran}}\right\|_F^2 + \lambda \|\mathbf{B}^{\mathrm{tran}}\|_F^2, \tag{6}$$

where $\mathbf{B}^{\mathrm{tran}}$ is the item-item relevance matrix learned from the sequential dependency between $\mathbf{Y}$ and $\mathbf{Z}$. As $\mathbf{Y} \neq \mathbf{Z}$, we can naturally avoid the trivial solution $\mathbf{B}^{\mathrm{tran}} = \mathbf{I}$. Unlike Eq. (2), we can remove the zero-diagonal constraint in $\mathbf{B}^{\mathrm{tran}}$. The closed-form solution is given by

$$\hat{\mathbf{B}}^{\mathrm{tran}} = \hat{\mathbf{P}}' \cdot (\mathbf{Y}^{\top}\mathbf{Z}), \tag{7}$$

where $\hat{\mathbf{P}}' = (\mathbf{Y}^{\top}\mathbf{Y} + \lambda \mathbf{I})^{-1} \in \mathbb{R}^n$. The computational complexity is independent of the number of users, as shown in Eq. (3). (See Appendix A for a detailed derivation of our solution in the supplementary material.)

To utilize the item transition matrix in random walks, each element should be the transition probability from one node to another. That is, every element is non-negative and sums to 1. However, $\hat{\mathbf{B}}^{\mathrm{tran}}$ is not normally a probability matrix. To satisfy the non-negative constraint, we first replace all negative values in $\hat{\mathbf{B}}^{\mathrm{tran}}$ with zero[1], denoting $\hat{\mathbf{B}}^{\mathrm{tran}}_{\geq 0}$. We then normalize $\hat{\mathbf{B}}^{\mathrm{tran}}_{\geq 0}$ as follows:

$$\mathbf{R} = \mathtt{diagMat}(\hat{\mathbf{B}}^{\mathrm{tran}}_{\geq 0}\mathbf{1})^{-1}\hat{\mathbf{B}}^{\mathrm{tran}}_{\geq 0}. \tag{8}$$

Here, $\mathbf{R}$ is the item transition probability matrix, where each row is normalized, and $\mathbf{1}$ is a column vector of length n filled with one.

[1] As the negative values in $\hat{\mathbf{B}}^{\mathrm{tran}}$ mean negative correlations among items, they are less important than positive correlations. In our empirical study, it is observed that this conversion does not harm the accuracy of the linear model.

3.3 Item Teleportation Model

The item teleportation matrix is designed to capture the item consistency within a session. For this reason, we focus on modeling co-occurrence between items. A session is treated as a set of items $\mathbf{s} = \{s_1, s_2, \ldots, s_{|s|}\}$, ignoring the order of items. By stacking m sessions, we build a binary session-item matrix $\mathbf{X}$. Note that the repeated items in the session are treated as a single item.

Given the matrix $\mathbf{X}$, we devise a linear teleportation model. It is formulated with the same input and output matrix as used in the existing linear models [39, 46]. Meanwhile, we relax the zero-diagonal constraint for $\mathbf{B}$ to handle repeated item consumption, as discussed in [5]. When the diagonal element of $\mathbf{B}$ is loosely penalized, it allows us to predict the same items as the next item repeatedly.

$$\underset{\mathbf{B}^{\text{tele}}}{\operatorname{argmin}} \left\| (\mathbf{X} - \mathbf{X} \cdot \mathbf{B}^{\text{tele}}) \right\|_F^2 + \lambda \|\mathbf{B}^{\text{tele}}\|_F^2, \quad \text{s.t. } \texttt{diag}(\mathbf{B}^{\text{tele}}) \leq \xi, \tag{9}$$

where $\mathbf{B}^{\text{tele}}$ is the item-item relevance matrix for item consistency, and ξ is the hyper-parameter to control diagonal constraints. When $\xi = 0$, it is equivalent to the zero-diagonal constraint for $\mathbf{B}^{\text{tele}}$. When $\xi = \infty$, there is no constraint on the diagonal elements of $\mathbf{B}^{\text{tele}}$. Note that the objective function of EASE^R [46] is a special case of Eq. (9), where $\mathbf{B}^{\text{tele}}$ with $\xi = 0$.

We can obtain the closed-form solution of $\mathbf{B}^{\text{tele}}$ as follows:

$$\hat{\mathbf{B}}^{\text{tele}} = \mathbf{I} - \hat{\mathbf{P}} \cdot \texttt{diagMat}(\gamma), \quad \gamma_j = \begin{cases} \lambda & \text{if } 1 - \lambda P_{jj} \leq \xi, \\ \frac{1-\xi}{P_{jj}} & \text{otherwise,} \end{cases} \tag{10}$$

where $\hat{\mathbf{P}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}$, and $\gamma \in \mathbb{R}^n$ is the vector used to check the diagonal constraint of $\mathbf{B}^{\text{tele}}$. Because of the inequality condition for the diagonal elements in $\mathbf{B}^{\text{tele}}$, γ_j is determined by $(1 - \lambda P_{jj})$. The closed-form solution may depend on ξ. If $\xi = 0$, γ_j is equal to $1/P_{jj}$, corresponding to $(1 \oslash \texttt{diag}(\hat{\mathbf{P}}))$. If $\xi = \infty$, the solution becomes $\hat{\mathbf{B}}^{\text{tele}} = \mathbf{I} - \lambda \hat{\mathbf{P}}$.

Similar to $\hat{\mathbf{B}}^{\text{tran}}$, the solution $\hat{\mathbf{B}}^{\text{tele}}$ does not satisfy the non-negativity and normalization conditions to be a probability matrix. We compute the item teleportation probability matrix $\mathbf{T}$ by replacing the negative values with zero and normalizing $\hat{\mathbf{B}}^{\text{tele}}_{\geq 0}$:

$$\mathbf{T} = \beta \left(\texttt{diagMat}(\hat{\mathbf{B}}^{\text{tele}}_{\geq 0} \mathbf{1})^{-1} \hat{\mathbf{B}}^{\text{tele}}_{\geq 0} \right) + (1 - \beta)\, \mathbf{I}, \tag{11}$$

where β is a hyper-parameter that controls the importance of the self-loop to guarantee the convergence of random walks. (See Appendix C for detailed analysis in the supplementary material.)

3.4 Random-walk Training and Inference

Training. To compute the stationary distribution of S-Walk, we utilize the power method [26]. Each proximity score corresponds to the limiting distribution using Eq. (4):

$$\begin{aligned} \mathbf{x}_{(\infty)} &= \alpha^\infty \mathbf{x}_{(0)} \mathbf{R}^\infty + \sum_{k=0}^{\infty} \alpha^k (1 - \alpha) \mathbf{x}_{(0)} \mathbf{T} \mathbf{R}^k \\ &\approx \mathbf{x}_{(0)} \sum_{k=0}^{\infty} \alpha^k (1 - \alpha) \mathbf{T} \mathbf{R}^k = \mathbf{x}_{(0)} \mathbf{M}. \end{aligned} \tag{12}$$

Here, $\mathbf{M} = \sum_{k=0}^{\infty} \alpha^k (1 - \alpha) \mathbf{T} \mathbf{R}^k$ is the trained item-item matrix by S-Walk. (Appendix B provides the detailed pseudo-code for computing the proximity score using the power method.)

Inference. Given a new session s_{new}, we represent a session vector $\mathbf{x}_{\text{new}}$ and compute the proximity score using $\mathbf{M}$. The proximity score for predicting the next item is finally given by $\mathbf{x}_{\text{new}} \mathbf{M}$. In this process, we decay the importance of items in $\mathbf{x}_{\text{new}}$ to rely more on recent consumption, similar to Eq. (5):

$$w_{\text{inf}}(i, s_{\text{new}}) = \exp\left(-\frac{|s_{\text{new}}| - p(i, s_{\text{new}})}{\delta_{\text{inf}}} \right), \tag{13}$$

where δ_{inf} is the hyper-parameter used to control the item weight decay, and $|s_{\text{new}}|$ is the length of the session s_{new}.

4 EXPERIMENTAL SETUP

Benchmark datasets. We use four public datasets collected from e-commerce and music streaming services: YooChoose[2] (YC), Diginetica[3] (DIGI), RetailRocket[4] (RR), and NowPlaying[5] (NOWP). Following existing studies [16, 31, 32, 53], we use single training-test split (single-split) datasets (*i.e.*, YC-1/4, DIGI1). To minimize the risk of random effects, we also split the datasets (*i.e.*, YC5, DIGI5, RR, NOWP) into five folds (five-split) which are contiguous in time, as used in recent extensive evaluation [33–35]. Following the convention [33–35], we discard the sessions with only one interaction and items that occur less than five times. Also, we split the training and test datasets chronologically and use a portion of the training set for validation, such that the validation set size is equal to that of the test dataset. (See Appendix D for detailed statistics of all benchmark datasets.)

Competing models. We compare our model to nine competing models. Among the non-neural models, we compare to AR [2], SR [25], STAN [11], and SLIST [5]. AR [2] is a simple model using association rules, and SR [25] is a Markov chain model together with association rules. STAN [11] is an improved model of SKNN [21], a session-based kNN algorithm. SLIST [5] is a linear model designed for SR. Among the neural models, we compare to NARM [31], STAMP [32], SR-GNN [53], NISER+ [13], and GCE-GNN [52]. Notably, SR-GNN [53] employs GNNs to consider complex sequential dependency of items. To overcome overfitting and popularity bias, NISER+ [13] uses normalized items and session embeddings on top of SR-GNN. Lastly, GCE-GNN [52] considers inter-session relationships by modeling item transitions over all sessions. (See Appendix E for the implementation details of baselines.) All source codes are available at https://github.com/jin530/SWalk.

Evaluation protocol and metrics. As in the common protocol [18, 31, 53], we use the *iterative revealing scheme*, which iteratively exposes an item from a session to the model, to reflect sequential user behavior throughout the entire session. We adopt several metrics to handle two scenarios: (i) To evaluate the next single item, we use *Hit Rate (HR)* and *Mean Reciprocal Rank (MRR)*, which have been widely used in existing studies [17, 32, 53]. HR and MRR measure the next item's existence and rank in the recommendation list, respectively.

[2] https://www.kaggle.com/chadgostopp/recsys-challenge-2015
[3] https://competitions.codalab.org/competitions/11161
[4] https://www.kaggle.com/retailrocket/ecommerce-dataset
[5] https://drive.google.com/drive/folders/1ritDnO_Zc6DFEU6UND9C8VCisT0ETVp5

(ii) To evaluate all subsequent items predicted, we use two common IR measures, *Recall (R@k)* and *Mean Average Precision (MAP@k)*, which measure the subsequent items' existence and rank in the recommendation list, respectively. We use $k = \{5, 10, 20, 50, 100\}$, where $k = 20$ is the default.

5 EVALUATION RESULTS

We evaluate the accuracy and efficiency of S-Walk by comparing it with competing models. Based on thorough and extensive evaluations, we make the following conclusions.

- S-Walk exhibits **state-of-the-art performance** on multiple datasets. For R@20, MAP@20, and HR@20, S-Walk consistently outperforms the other models, up to 3.16%, 4.11%, and 3.65%, respectively (Section 5.1).
- The inference of S-Walk is up to **8.9 times faster** than other DNN-based models. Surprisingly, S-Walk can be **compressed** (sparsified by zeroing-out entries via thresholding), **without sacrificing its accuracy** (Section 5.2).
- For **long sessions**, where user intent is more difficult to capture, S-Walk significantly outperforms existing models. (Section 5.3).
- S-Walk **converges within 3–5 steps** and is superior to the k-step method because it utilizes the teleportation model for restarts (Section 5.4).

5.1 Evaluation of Accuracy

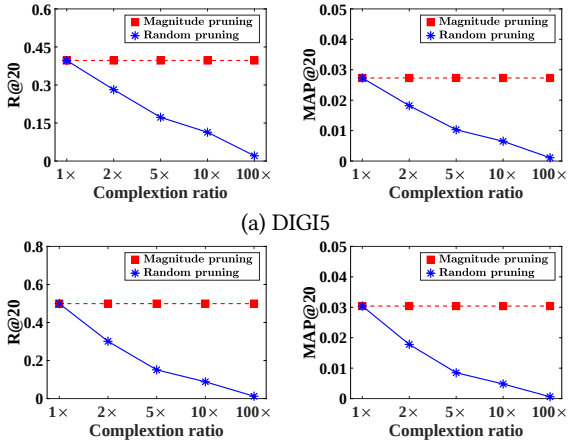
Table 1 reports the accuracy of S-Walk and other competitors. It is found that no single model shows the best performance over all the datasets, as reported in existing studies [33–35]. Remarkably, the existing models have different tendencies for single-split and five-split datasets; while most neural models surpass non-neural models on the single-split datasets, their performances are degraded on the five-split datasets. We conjecture that the parameters of the neural models reported in existing studies are mostly biased for optimizing the single-split datasets. For five-split datasets, it is also observed that the variance in the neural models is larger than that of non-neural models. Based on these observations, it is challenging for one model to consistently achieve outstanding performance on all the datasets.

Nevertheless, S-Walk shows competitive or state-of-the-art performances. Notably, the gain of S-Walk is up to 4.08% and 8.15% in R@20 and MAP@20 over the best competing model. For all five-split datasets, S-Walk consistently surpasses the existing models for HR@20, R@20, and MAP@20. These empirical results indicate that S-Walk captures various intra- and inter-correlation among items without being biased to specific datasets.

Compared to the other metrics, S-walk is slightly lower on MRR@20, particularly on YC and NOWP datasets. Based on the gap between SR [25] and STAN [11], we observe that the MRR@20 scores on these datasets are mostly affected by intra-session relationships. Thus, we address this by taking fewer random walks on these datasets; *e.g.*, S-Walk$_{(1)}$ considers mostly intra-session relationships, confirming superior performance in MRR@20 as well.

5.2 Evaluation of Scalability

Table 2 compares the inference time between S-Walk and the other best models on several datasets. Whereas the computational cost of S-Walk is proportional only to the number of items, that of neural models also depends on the number of layers and their dimensions. Although S-Walk runs on CPU and other models run on GPU, S-Walk shows about five times faster inference time owing to its simpler structure, even with better accuracy. This property is highly desirable for deploying S-Walk for real-world applications.



(a) DIGI5

(b) RR

Figure 3: R@20 and MAP@20 of S-Walk over various compression ratios.

We also attempt *model pruning* for S-Walk. As a simple strategy, we adopt magnitude pruning [10], *i.e.*, the model is globally sparsified by zeroing out parameter values with the smallest magnitude. For instance, 100× compression means that we retain only 1% non-zero entries in $\mathbf{M}$ with the highest magnitude while zeroing out the remaining 99%. Surprisingly, S-Walk preserves the accuracy with highly extreme (100×) compression ratios, as depicted in Figure 3. Random pruning is a simple baseline, *i.e.*, the parameters are randomly removed under the given compression ratio. This result indicates that the learned relationship between items is disentangled with the locality, as observed in existing studies [4, 27–30]. Owing to this valuable property, the compressed S-Walk is memory-efficient, suitable for low-resource devices, *e.g.*, mobile and embedded applications.

5.3 Effect of Various Session Data

To further investigate the accuracy gains of S-Walk we carefully design case studies and examine the effects of session length and retrieval size. (See Appendix F for the effect of the data size on S-Walk and competing models.)

To observe how the session length affects the performance, we compare it with other best models (*i.e.*, STAN [11], SR-GNN [53] and NISER+ [13]) by categorizing the entire sessions into two groups: short ($\leq$ 5 items) and long sessions ($>$ 5 items). Note that the ratio of short sessions is 77.2% (DIGI5) and 79.2% (RR), respectively. Figure 4 indicates that long sessions are much more challenging to predict users' hidden intents effectively. Nonetheless, S-Walk significantly outperforms NISER+ [13] in long sessions by 6.2%–20.2% and 5.8%–21.0% on R@20 and MAP@20, respectively. Based on this result, we

Table 1: Accuracy comparison of S-Walk and competing models, following experimental setup in [34, 35]. Gains indicate how much better the best proposed model is than the best baseline model. S-Walk$_{(1)}$ is a variant of S-Walk trained only up to the first step $M_{(1)}$. The best model is marked in bold and the second best model is underlined.

Dataset	Metric	Non-Neural Models				Neural Models					Ours		Gain(%)
		AR	SR	STAN	SLIST	NARM	STAMP	SR-GNN	NISER+	GCE-GNN	S-Walk$_{(1)}$	S-Walk	
DIGI5	R@20	0.2872	0.2517	0.3720	0.3803	0.3254	0.3040	0.3232	0.3727	0.3927	0.3761	**0.3995**	1.73
	MAP@20	0.0189	0.0164	0.0252	0.0259	0.0218	0.0201	0.0217	0.0259	0.0271	0.0254	**0.0277**	2.21
	HR@20	0.3720	0.3277	0.4800	0.4915	0.4188	0.3917	0.4158	0.4785	0.5086	0.4873	**0.5115**	0.57
	MRR@20	0.1280	0.1216	0.1828	0.1809	0.1392	0.1314	0.1436	0.1656	0.1803	0.1732	**0.1890**	3.39
RR	R@20	0.3533	0.3359	0.4748	0.4724	0.4526	0.3917	0.4438	0.4630	0.4841	0.4810	**0.4994**	3.16
	MAP@20	0.0205	0.0194	0.0285	0.0282	0.0270	0.0227	0.0264	0.0278	0.0292	0.0290	**0.0304**	4.11
	HR@20	0.4367	0.4174	0.5938	0.5877	0.5549	0.4620	0.5433	0.5651	0.6007	0.6019	**0.6226**	3.65
	MRR@20	0.2407	0.2453	0.3638	0.3131	0.3196	0.2527	0.3066	0.3171	0.3362	0.3409	**0.3645**	0.19
YC5	R@20	0.4760	0.4853	0.4986	0.5122	0.5109	0.4979	0.5060	0.5146	0.4972	0.5096	**0.5189**	0.85
	MAP@20	0.0325	0.0332	0.0342	0.0357	0.0357	0.0344	0.0351	0.0363	0.0348	0.0355	**0.0365**	0.55
	HR@20	0.6361	0.6506	0.6656	0.6867	0.6751	0.6654	0.6713	0.6858	0.6650	0.6834	**0.6906**	0.57
	MRR@20	0.2894	0.3010	0.2933	0.3097	0.3047	0.3033	**0.3142**	0.3130	0.2939	0.3074	0.2892	-2.17
NOWP	R@20	0.1544	0.1366	0.1696	0.1840	0.1274	0.1253	0.1400	0.1493	0.1504	0.1837	**0.1915**	4.08
	MAP@20	0.0166	0.0133	0.0175	0.0184	0.0118	0.0113	0.0125	0.0143	0.0143	0.0182	**0.0199**	8.15
	HR@20	0.2076	0.2002	0.2414	0.2689	0.1849	0.1915	0.2113	0.2196	0.2122	0.2678	**0.2693**	0.15
	MRR@20	0.0710	0.1052	0.0871	0.1137	0.0894	0.0882	0.0935	0.0944	0.0720	**0.1158**	0.0866	1.85
DIGI1	R@20	0.3401	0.3164	0.3965	0.4091	0.3890	0.3663	0.3811	**0.4202**	0.4169	0.4012	0.4201	-0.02
	MAP@20	0.0231	0.0212	0.0275	0.0286	0.0270	0.0252	0.0264	**0.0299**	0.0295	0.0278	0.0297	-0.67
	HR@20	0.4394	0.4085	0.5121	0.5291	0.4979	0.4690	0.4896	0.5397	0.5362	0.5188	**0.5420**	0.43
	MRR@20	0.1465	0.1431	0.1851	0.1886	0.1585	0.1499	0.1654	0.1856	0.1907	0.1869	**0.1927**	1.05
YC-1/4	R@20	0.4781	0.4851	0.4952	0.5130	0.5097	0.5008	0.5095	0.5164	0.5030	0.5103	**0.5213**	0.95
	MAP@20	0.0363	0.0364	0.0373	0.0393	0.0395	0.0385	0.0393	0.0402	0.0388	0.0390	**0.0404**	0.50
	HR@20	0.6697	0.6850	0.6846	0.7175	0.7079	0.7021	0.7118	0.7182	0.7036	0.7145	**0.7204**	0.31
	MRR@20	0.2880	0.3053	0.2829	0.3161	0.2996	0.3066	**0.3180**	0.3161	0.3027	0.3137	0.2867	-1.35

Table 2: The number of floating point operations and runtime (in seconds) for inference of S-Walk and DNN-based models. Gain indicate how fast S-Walk is compared to GCE-GNN [52]. Runtime for DNN models was measured on GPU, while that for S-Walk on CPU.

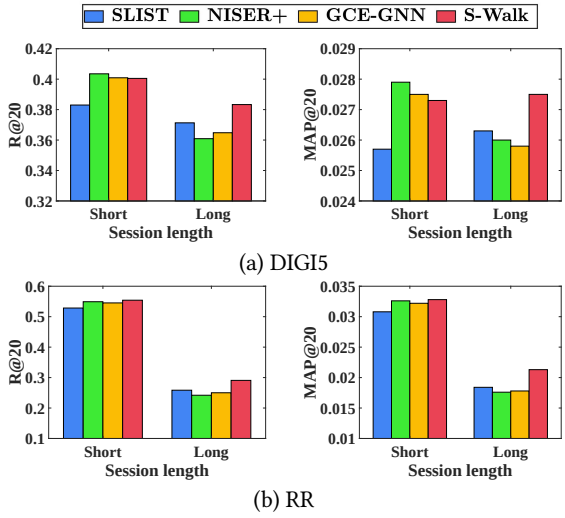
Models	YC1/4		DIGI5		RR	
	GFLOPs	Time	GFLOPs	Time	GFLOPs	Time
SR-GNN	1282.8	70.8	765.4	49.2	247.2	12.7
NISER+	2605.8	87.1	1551.0	59.7	501.8	15.7
GCE-GNN	51094.8	108.8	10445.9	74.0	9446.0	19.8
S-Walk	11.0	20.5	4.9	8.3	2.3	5.2
Gain	4632.3x	5.3x	2131.3x	8.9x	4133.2x	3.8x

confirm that S-Walk effectively captures complicated item patterns, further improving the accuracy of longer sessions.

Figure 5 depicts the comparison results between S-Walk and the competitive models for various numbers of recommended items. For all cut-off sizes, we observe that S-Walk consistently surpasses NISER+ by 5.2%–16.9% and 5.4%–17.2% gains on Recall and MAP, respectively. In this sense, S-Walk can expand item coverage by increasing the number of steps through the random walk process in the item graph.

5.4 Ablation Study

We analyze the effect of each component, *i.e.*, the transition model and the teleportation model, in S-Walk. As shown in Table 3, the



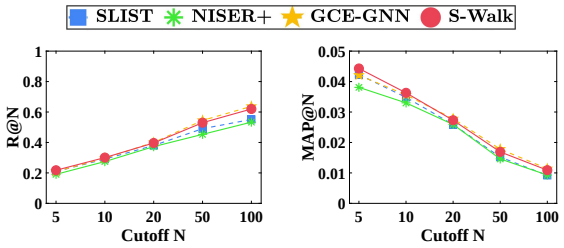
(a) DIGI5

(b) RR

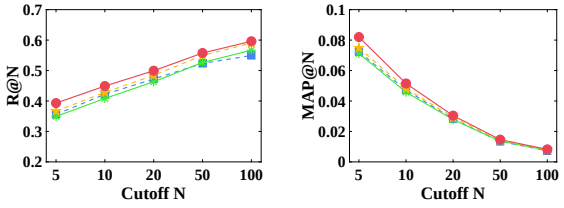
Figure 4: R@20 and MAP@20 of S-Walk and competitor models over different session lengths (Short ≤ 5, Long > 5).

complete S-Walk shows the best performance, compared to using SR [25] for the transition model. For the teleportation model, AR [2] shows worse accuracy than the identity matrix in the YC-1/4 dataset, where AR [2] shows the worst performance. This implies that incorrect teleportation may hinder random walks.

Figure 6 depicts the effect of the damping factor α in S-Walk, controlling the ratio of walks and restarts. With $\alpha = 0.7$, S-Walk



(a) DIGI5

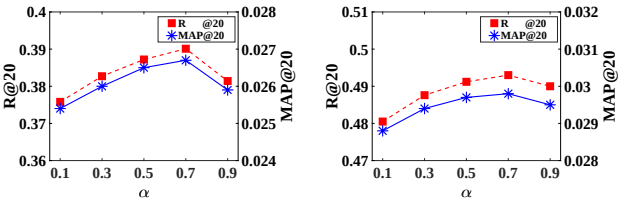


(b) RR

Figure 5: R@N and MAP@N of S-Walk and state-of-the-art models over the various cut-offs on DIGI5 and RR.

Table 3: R@20 and MAP@20 of S-Walk with various transition and teleportation models. SR [25] and AR [2] are simple Markov- and co-occurence-based models. I denotes the identity matrix. *tran* and *tele* denote the transition and teleportation model, respectively.

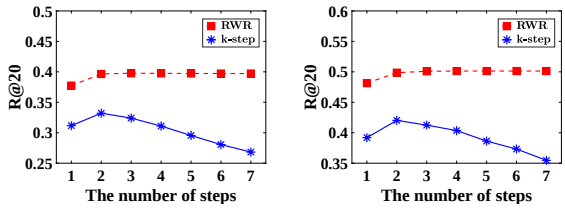
tran	*tele*	YC-1/4		DIGI5		RR	
		R	MAP	R	MAP	R	MAP
SR	I	0.5109	0.0394	0.3809	0.0260	0.4812	0.0291
	AR	0.4952	0.0378	0.3879	0.0266	0.4817	0.0291
	Ours	0.5171	0.0400	0.3930	0.0270	0.4950	0.0301
Ours	I	0.5175	0.0399	0.3808	0.0259	0.4826	0.0292
	AR	0.5009	0.0383	0.3899	0.0268	0.4856	0.0293
	Ours	**0.5205**	**0.0403**	**0.3936**	**0.0271**	**0.4979**	**0.0303**



(a) DIGI5 (b) RR

Figure 6: R@20 and MAP@20 of S-Walk over varied damping factor α.

shows the best performance, implying that the transition model is more dominant than the teleportation model. Finally, with various numbers of random walk steps, we compare S-walk with the k-step method, which uses the k-step landing probability on the transition graph. (i) Without restart using the teleportation graph, performance significantly degrades, and (ii) when $k \geq 2$ in k-step, the accuracy continues to decrease. (iii) S-Walk usually converges within 3–5 steps, achieving the best performance on both datasets.



(a) DIGI5 (b) RR

Figure 7: R@20 of S-Walk using random walk with restart (RWR) and a k-step landing probability.

6 RELATED WORK

Random walk-based models. With the success of PageRank [26], the idea of the random walk has been widely adopted to address the data sparsity problem in recommender systems [55]. TrustWalker [20] addressed cold-start user/item problems using trust information between users. Starting from a target user vertex, [6, 7] estimated the proximity score using the transition probabilities after short random walks on the user-item bipartite network. Besides, random-walk-based models have been successfully deployed in the large-scale industrial systems [8]. Recently, RecWalk [37, 38] leveraged spectral properties of nearly decoupled Markov chains and combined the item model with random walks.

Session-based recommendation (SR). SR models are categorized into three groups, *i.e.*, *Markov chain models*, *neighborhood-based models*, and *DNN-based models*. For more details, please refer to recent survey papers [3, 22]. Firstly, Markov chains (MC) are useful for modeling consecutive item dependency. FPMC [43] proposed the tensor factorization using MF, and FOSSIL [14] combined FISM [24] with factorized MC. SR [25] proposed an improved MC model by combining association rules. Although they are effective for addressing the short-term item dependency, it does not utilize various patterns among items. Secondly, Jannach and Ludewig [21] adopted the K-nearest neighbor (KNN) for SR, and STAN [11] improved SKNN using various weight schemes to further reflect item dependency. Recently, [33–35] reported that the KNN-based models have shown competitive performance on various datasets. However, they are generally limited in representing high-order dependency among items. Lastly, various DNN-based models have been used for SR. GRU4Rec [16, 17] employed gated recurrent units (GRU) for SR. Later, NARM [31] and STAMP [32] utilized attention mechanisms to distinguish short- and long-term item dependency. To further analyze complex item transitions, SR-GNN [53] recently exploited gated graph neural networks (GGNN). However, SR-GNN [53] is often vulnerable to overfitting [13] owing to extreme data sparsity.

7 CONCLUSION

In this work, we propose a session-based recommendation using item random walks, *S-Walk*. To complement the drawback of existing models, we utilize the random walk with restart to fully capture intra- and inter-correlations of sessions. We incorporate efficient linear item models, *i.e.*, the transition model and the teleportation model, into the item random walks process. Our extensive evaluation shows that S-Walk achieves comparable or state-of-the-art accuracies, high scalability, and fast inference speed over various benchmark datasets.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) (NRF-2018R1A5A1060031, NRF-2021R1F1A1063843, and NRF-2021H1D3A2A03038607). Also, this work was supported by Institute of Information & communications Technology Planning & evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00421, AI Graduate School Support Program).

REFERENCES

[1] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. 2020. N-GCN: Multi-scale graph convolution for semi-supervised node classification. In *Uncertainty in Artificial Intelligence (UAI)*.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD*. 207–216.

[3] Geoffray Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. *ACM Comput. Surv.* 47, 2 (2014), 26:1–26:35.

[4] Minjin Choi, Yoonki Jeong, Joonseok Lee, and Jongwuk Lee. 2021. Local Collaborative Autoencoders. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 734–742.

[5] Minjin Choi, Jinhong Kim, Joonseok Lee, Hyunjung Shim, and Jongwuk Lee. 2021. Session-aware Linear Item-Item Models for Session-based Recommendation. In *WWW*. 2186–2197.

[6] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. 2015. Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks. In *RecSys*. 163–170.

[7] Colin Cooper, Sang-Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: exact computation and simulations. In *WWW (Companion Volume)*. 811–816.

[8] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. In *WWW*. 1775–1784.

[9] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. 2019. Deep Learning-Based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations. In *ICWE*. 574–577.

[10] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *ICLR*.

[11] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN. In *SIGIR*. 1069–1072.

[12] David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35, 12 (1992), 61–70.

[13] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. NISER: Normalized Item and Session Representations with Graph Neural Networks. *CoRR* (2019).

[14] Ruining He and Julian J. McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *ICDM*. 191–200.

[15] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *SIGIR*. 230–237.

[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *CIKM*. 843–852.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[18] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys*. 241–248.

[19] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. 263–272.

[20] Mohsen Jamali and Martin Ester. 2009. *TrustWalker*: a random walk model for combining trust-based and item-based recommendation. In *KDD*. 397–406.

[21] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *RecSys*. 306–310.

[22] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. *User Model. User Adapt. Interact.* 27, 3-5 (2017), 351–392.

[23] Olivier Jeunen, Jan Van Balen, and Bart Goethals. 2020. Closed-Form Models for Collaborative Filtering with Side-Information. In *RecSys*. 651–656.

[24] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *KDD*. 659–667.

[25] Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. 2017. A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation. In *RecSys*. 50–56.

[26] Amy N. Langville and Carl D. Meyer. 2006. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, USA.

[27] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *WWW*. 85–96.

[28] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *ICML*. 82–90.

[29] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Matrix approximation under local low-rank assumption. In *ICLR*.

[30] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. 2016. LLORMA: Local Low-Rank Matrix Approximation. *Journal of Machine Learning Research* 17 (2016), 15:1–15:24.

[31] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*. 1419–1428.

[32] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD*. 1831–1839.

[33] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Model. User Adapt. Interact.* 28, 4-5 (2018), 331–390.

[34] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Empirical Analysis of Session-Based Recommendation Algorithms. *CoRR* abs/1910.12781 (2019).

[35] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance comparison of neural and non-neural approaches to session-based recommendation. In *RecSys*. 462–466.

[36] Anjing Luo, Pengpeng Zhao, Yanchi Liu, Fuzhen Zhuang, Deqing Wang, Jiajie Xu, Junhua Fang, and Victor S. Sheng. 2020. Collaborative Self-Attention Network for Session-based Recommendation. In *IJCAI*. 2591–2597.

[37] Athanasios N. Nikolakopoulos and George Karypis. 2019. RecWalk: Nearly Uncoupled Random Walks for Top-N Recommendation. In *WSDM*. 150–158.

[38] Athanasios N. Nikolakopoulos and George Karypis. 2020. Boosting Item-based Collaborative Filtering via Nearly Uncoupled Random Walks. *ACM Trans. Knowl. Discov. Data* 14, 6 (2020), 64:1–64:26.

[39] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *ICDM*. 497–506.

[40] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. 2020. Star Graph Neural Networks for Session-based Recommendation. In *CIKM*. 1195–1204.

[41] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In *SIGIR*. 1837–1840.

[42] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4 (2018), 66:1–66:36.

[43] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. 811–820.

[44] Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2015. *Recommender Systems Handbook*. Springer.

[45] Harald Steck. 2019. Collaborative Filtering via High-Dimensional Regression. *CoRR* abs/1904.13033 (2019).

[46] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *WWW*. 3251–3257.

[47] Harald Steck. 2019. Markov Random Fields for Collaborative Filtering. In *NeurIPS*. 5474–5485.

[48] Koen Verstrepen and Bart Goethals. 2014. Unifying nearest neighbors collaborative filtering. In *RecSys*. 177–184.

[49] Maksims Volkovs and Guang Wei Yu. 2015. Effective Latent Models for Binary Feedback in Recommender Systems. In *SIGIR*. 313–322.

[50] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In *SIGIR*. 345–354.

[51] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. *arXiv:1902.04864* (2019).

[52] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2020. Global Context Enhanced Graph Neural Networks for Session-based Recommendation. In *SIGIR*. 169–178.

[53] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *AAAI*. 346–353.

[54] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*. 3940–3946.

[55] Hilmi Yildirim and Mukkai S. Krishnamoorthy. 2008. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys*. 131–138.

A CLOSED-FORM SOLUTIONS

We provide a detailed derivation of the closed-form solutions for the item transition model and the item teleportation model in Section 3.2 and 3.3. Based on the closed-form solutions, we can further apply for row-level weights to reflect the *timeliness* of sessions, *i.e.*, the more important, the more recent sessions.

A.1 Linear Item Transition Model

Given the input matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$ and the output matrix $\mathbf{Z} \in \mathbb{R}^{m \times n}$, the objective function is expressed as

$$\underset{\mathbf{B}^{\text{tran}}}{\operatorname{argmin}} \ \mathcal{L}(\mathbf{B}^{\text{tran}}) = \left\|(\mathbf{Z} - \mathbf{Y} \cdot \mathbf{B}^{\text{tran}})\right\|_F^2 + \lambda \|\mathbf{B}^{\text{tran}}\|_F^2. \tag{14}$$

The first-order derivative of Eq. (14) over $\mathbf{B}^{tran}$ is then given by

$$\begin{aligned} \frac{1}{2} \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{B}^{\text{tran}}} &= (-\mathbf{Y}^\top)(\mathbf{Z} - \mathbf{Y} \cdot \mathbf{B}^{\text{tran}}) + \lambda \mathbf{B}^{\text{tran}}, \\ &= (\mathbf{Y}^\top \mathbf{Y} + \lambda \mathbf{I}) \cdot \mathbf{B}^{\text{tran}} - \mathbf{Y}^\top \mathbf{Z}. \end{aligned} \tag{15}$$

Solving for $\mathbf{B}^{\text{tran}}$ so that Eq. (15) becomes 0 gives the closed-form solution of Eq. (14):

$$\hat{\mathbf{B}}^{\text{tran}} = \hat{\mathbf{P}}' \cdot (\mathbf{Y}^\top \mathbf{Z}), \tag{16}$$

where $\hat{\mathbf{P}}' = \left(\mathbf{Y}^\top \mathbf{Y} + \lambda \mathbf{I}\right)^{-1}$.

A.2 Linear Item Teleportation Model

To solve the constrained optimization problem, we define a new objective function $\mathcal{L}(\mathbf{B}^{\text{tele}}, \mu)$ by applying a Lagrangian multiplier and a KKT condition:

$$\begin{aligned} \mathcal{L}(\mathbf{B}^{\text{tele}}) &= \|(\mathbf{X} - \mathbf{X} \cdot \mathbf{B}^{\text{tele}})\|_F^2 + \lambda \|\mathbf{B}^{\text{tele}}\|_F^2 \\ &\text{s.t. } \texttt{diag}(\mathbf{B}^{\text{tele}}) \le \xi \end{aligned} \tag{17}$$

$$\begin{aligned} \mathcal{L}(\mathbf{B}^{\text{tele}}, \mu) &= \|(\mathbf{X} - \mathbf{X} \cdot \mathbf{B}^{\text{tele}})\|_F^2 + \lambda \|\mathbf{B}^{\text{tele}}\|_F^2 \\ &+ 2\mu^\top \texttt{diag}(\mathbf{B}^{\text{tele}} - \xi \mathbf{I}), \end{aligned} \tag{18}$$

where $\mu \in \mathbb{R}^n$ is the KKT multiplier that satisfies $\forall i, \mu_i \ge 0$. Then, we differentiate $\mathcal{L}(\mathbf{B}^{\text{tele}}, \mu)$ with respect to $\mathbf{B}^{\text{tele}}$ to minimize Eq. (18):

$$\begin{aligned} \frac{1}{2} \frac{\partial \mathcal{L}(\mathbf{B}^{\text{tele}}, \mu)}{\partial \mathbf{B}^{\text{tele}}} &= (-\mathbf{X}^\top)(\mathbf{X} - \mathbf{X} \cdot \mathbf{B}^{\text{tele}}) + \lambda \mathbf{B}^{\text{tele}} + \texttt{diagMat}(\mu) \\ &= -\mathbf{X}^\top \mathbf{X} + \mathbf{X}^\top \mathbf{X} \cdot \mathbf{B}^{\text{tele}} + \lambda \mathbf{B}^{\text{tele}} + \texttt{diagMat}(\mu) \\ &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}) \mathbf{B}^{\text{tele}} - \mathbf{X}^\top \mathbf{X} + \texttt{diagMat}(\mu). \end{aligned} \tag{19}$$

Setting this to 0 and solving by $\mathbf{B}^{\text{tele}}$ gives the optimal $\hat{\mathbf{B}}^{\text{tele}}$ as

$$\begin{aligned} \hat{\mathbf{B}}^{\text{tele}} &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \cdot [\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I} - \lambda \mathbf{I} - \texttt{diagMat}(\mu)] \\ &= \hat{\mathbf{P}}[\hat{\mathbf{P}}^{-1} - \lambda \mathbf{I} - \texttt{diagMat}(\mu)] \\ &= \mathbf{I} - \hat{\mathbf{P}}[\lambda \mathbf{I} + \texttt{diagMat}(\mu)] \\ &= \mathbf{I} - \lambda \hat{\mathbf{P}} - \hat{\mathbf{P}} \cdot \texttt{diagMat}(\mu), \end{aligned} \tag{20}$$

where $\hat{\mathbf{P}} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1}$.

Besides, a KKT multiplier μ_j is zero only if $\mathbf{B}_{jj}^{\text{tele}} \le \xi$. Otherwise, μ_j has a non-zero value. In this case, μ_i regularizes the value of $\mathbf{B}_{jj}^{\text{tele}}$ as $\mathbf{B}_{jj}^{\text{tele}} = \xi$. For $\mathbf{B}_{jj}^{\text{tele}}$, we can develop the the following equation:

$$\mathbf{B}_{jj}^{\text{tele}} = \xi = 1 - \lambda P_{jj} - P_{jj} \mu_j. \tag{21}$$

Finally, μ_j can be expressed as follows:

$$\mu_j = \frac{(1 - \lambda P_{jj} - \xi)}{P_{jj}} = \frac{(1 - \xi)}{P_{jj}} - \lambda. \tag{22}$$

Substituting μ in Eq. (20) and enforcing non-negative elements in $\hat{\mathbf{B}}^{\text{tele}}$ give $\hat{\mathbf{B}}^{\text{tele}}$:

$$\hat{\mathbf{B}}^{\text{tele}} = \mathbf{I} - \hat{\mathbf{P}} \cdot \texttt{diagMat}(\gamma), \tag{23}$$

$$\gamma_j = \begin{cases} \lambda & \text{if } \ 1 - \lambda P_{jj} \le \xi \\ \frac{1-\xi}{P_{jj}} & \text{otherwise.} \end{cases} \tag{24}$$

Here, γ is a vector defined by $\gamma = \mu + \lambda \cdot \mathbf{1}$.

B TRAINING OF S-WALK

Algorithm 1 describes how the final item-item matrix $\mathbf{M}$ is trained using the transition matrix $\mathbf{R}$ and the teleportation matrix $\mathbf{T}$. First, we define $\mathbf{M}_{(0)}$ as the identity matrix. Then, we repeatedly update $\mathbf{M}_{(k)}$ using $\mathbf{R}$ and $\mathbf{T}$. For example, $\mathbf{M}_{(1)}$ can be thought as $\mathbf{R}$ with a probability of α and $\mathbf{T}$ with a probability of $(1 - \alpha)$. After $\mathbf{M}_{(k)}$ converges, we use $\mathbf{M}_{(k)}$ as the item-item matrix for inference.

Algorithm 1: Training procedure for S-Walk

Input: Transition matrix $\mathbf{R}$, teleportation matrix $\mathbf{T}$.
Output: S-Walk model $\mathbf{M}$.

$\mathbf{M}_{(0)} \leftarrow \mathbf{I}$
$k = 0$
repeat
  $k \leftarrow k + 1$
  $\mathbf{M}_{(k)} \leftarrow \alpha \mathbf{M}_{(k-1)} \mathbf{R} + (1 - \alpha) \mathbf{T}$
until $\|\mathbf{M}_{(k)} - \mathbf{M}_{(k-1)}\|_1 \le \epsilon$
$\mathbf{M} \leftarrow \mathbf{M}_{(k)}$

C THEORETICAL ANALYSIS

Property of the item transition matrix. The item transition matrix $\mathbf{R}$ can be viewed as a generalized Markov model. When representing partial representations, the importance of items can be different. As δ_{pos} in Eq. (5) is small, it can be close to a simple Markov model [25], capturing consecutive transitions between items. We can also consider a broader range to represent the transition between items. While the simple Markov model is mostly based on the frequency of consecutive items, our linear model can estimate the sequential correlation between items. As a result, it can outperform existing Markov models [14, 24, 25, 43].

Property of the item teleportation matrix. As discussed in [46], assume that a training matrix $\mathbf{X}$ is $|\mathcal{S}|$ samples with $|\mathcal{I}|$ random variables, *i.e.*, $x \sim \mathcal{N}(0, \Sigma)$ following a Gaussian distribution with zero mean and the covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. Here, the estimate of the covariance matrix is $\hat{\Sigma} = \mathbf{X}^\top \mathbf{X} / |\mathcal{S}|$, and $\hat{\mathbf{P}} = \hat{\Sigma}^{-1}$ is the estimate of the precision (or concentration) matrix. By solving the closed-form equation in Eq. (10), the item teleportation matrix $\mathbf{T}$ can be interpreted as the precision matrix by estimating the co-occurrence matrix $\mathbf{X}^\top \mathbf{X} / |\mathcal{S}|$. In other words, the precision matrix can be viewed as the similarity matrix between items, which has been used in the recent neighborhood-based approach [48, 49].

Table 4: Statistics of the benchmark datasets. #Actions indicates the number of entire user-item interactions.

Split	Dataset	#Actions	#Sessions	#Items	#Actions / Sess.	#Items / Sess.
1-split	YC-1/4	7,909,307	1,939,891	30,638	4.08	3.28
	DIGI1	916,370	188,807	43,105	4.85	4.08
5-split	YC5	5,426,961	1,375,128	28,582	3.95	3.17
	DIGI5	203,488	41,755	32,137	4.86	4.08
	RR	212,182	59,962	31,968	3.54	2.56
	NOWP	271,177	27,005	75,169	10.04	9.38

Model convergence. The graph $\mathcal{G}_{\mathbf{M}}$ is defined by a combination of $\mathbf{R}$ and $\mathbf{T}$. The non-negative values in both matrices imply the positive correlations in covariance matrix $\mathbf{R}^{\top}\mathbf{T}$. The edges based on the correlations make graph $\mathcal{G}_{\mathbf{M}}$. Besides, $\mathbf{T}$ has a self-loop connection as in Eq. (11). Based on these structures, the transition matrix $\mathbf{R}$ is an ergodic Markov model, *i.e.*, irreducible and aperiodic. That is, the landing probabilities of S-Walk converge to a limiting distribution. It is found that S-Walk converges by 3–5 steps, as observed in 5.4.

D DATASET DESCRIPTION

Table 4 summarizes the detailed statistics of all the benchmark datasets. These datasets are collected from e-commerce and music streaming services: YooChoose[6] (YC), Diginetica[7] (DIGI), RetailRocket[8] (RR), and NowPlaying[9] (NOWP). For YC and DIGI, we use single-split datasets (*i.e.*, YC-1/4 and DIGI1), following existing studies [16, 31, 32, 53]. To evaluate on large-scale datasets, we further experiment on five-split datasets (*i.e.*, YC, DIGI5, RR, NOWP), used in the recent empirical analysis [33–35]. For single-split datasets, we use the last day as the test set for YC-1/4 and the sessions of the last seven days as the test set for DIGI1, as done in [33–35]. For five-split datasets, we divide the datasets into five disjoint successive splits. For each split, the last N-days sessions were used for the test set. We choose N for the test set as follows: one for YC, two for RR, five for NOWP, and seven for DIGI.

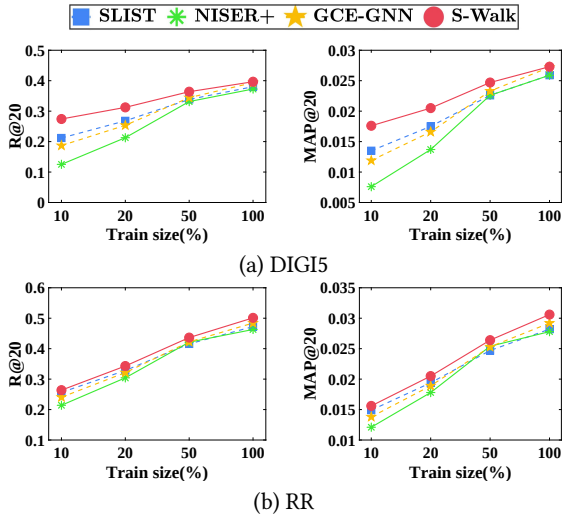
E REPRODUCIBILITY

Implementation details. For S-Walk, we tuned α and β among {0.1, 0.3, 0.5, 0.7, 0.9}, δ_{pos} and δ_{inf} among {0.125, 0.25, 0.5, 1, 2, 4, 8} using the validation set selected from the training set for the same period as the testing set. For the baseline models, we used the best hyper-parameters reported in [33, 34]. We implemented the proposed model and STAN [11] using NumPy. We used public source code for SR-GNN[10] [53] and implemented NISER+ [13] using SR-GNN code. We used all source code for the other baseline models[11] in [35]. For reproducibility, we conducted the reported results for all the baseline models, and actually verified whether the performance of baseline models could be reproduced with an error of 1–2% or less in our implementation environments. We conducted all experiments on a desktop with 2 NVidia TITAN RTX, 256 GB memory, and 2 Intel Xeon Processor E5-2695 v4 (2.10 GHz, 45M cache).

[6]https://www.kaggle.com/chadgostopp/recsys-challenge-2015
[7]https://competitions.codalab.org/competitions/11161
[8]https://www.kaggle.com/retailrocket/ecommerce-dataset
[9]https://drive.google.com/drive/folders/1ritDnO_Zc6DFEU6UND9C8VCisT0ETVp5
[10]https://github.com/CRIPAC-DIG/SR-GNN
[11]https://github.com/rn5l/session-rec

Table 5: Hyper-parameter settings of S-Walk. λ is a L2-weight decay, α is a damping factor and β is the self-loop factor. δ_{pos} is a weight decay by item position and δ_{inf} is a weight decay by inference item position.

Dataset	α	β	λ	δ_{pos}	δ_{inf}
YC-1/4	0.5	0.7	10	1	1
DIGI1	0.5	0.9	10	0.5	2
YC5	0.5	0.7	10	1	1
DIGI5	0.7	0.7	10	0.5	4
RR	0.5	0.9	10	0.25	4
NOWP	0.5	0.9	10	1	1



(a) DIGI5

(b) RR

Figure 8: R@20 and MAP@20 of S-Walk and competing models over various size of training session matrices.

Hyper-parameter setting. Table 5 reports all hyper-parameters for S-Walk. Firstly, the hyper-parameters of the left column are the factor that controls the involvement of random walk models. α is a damping factor that determines the proportion of the transition matrix $\mathbf{S}$ and teleportation matrix $\mathbf{T}$. β controls the probability that the surfer jumps to the item rather than to itself. Lastly, the hyper-parameters of the right column are model hyper-parameters used for the transition model and the teleportation model.

F ADDITIONAL EXPERIMENTAL RESULTS

We also evaluate the accuracy of S-Walk and the competing models by varying the size of the entire session from 10–100%. As shown in Figure 8, S-Walk consistently achieves better accuracy than other baselines. Even though all models degrade performance with a smaller training set, S-Walk suffers the least from sparser data. Notably, DNN-based models show significant degradation in accuracy with smaller training sets owing to their complex structures and enormous parameters. (For the other datasets, we still observe similar tendencies.) Based on this observation, S-Walk is still more effective than the baseline models, even on small-scale datasets.